Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №2 по курсу

«Операционные системы»

Группа: М8О-209БВ-24

Студент: Андреев М.В

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 05.11.25

Москва, 2025

Постановка задачи

Вариант 18.

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска вашей программы.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

Найти образец в строке наивным алгоритмом

Общий метод и алгоритм решения

Использованные системные вызовы:

- clock_gettime(CLOCK_MONOTONIC, &start) - получение монотонного времени для точного замера производительности
- pthread_create() - создание потоков для параллельного поиска
- pthread_join() - ожидание завершения всех потоков
- pthread_mutex_lock() / pthread_mutex_unlock() - синхронизация доступа к общему массиву результатов

Алгоритм работы программы:

1. **Инициализация и распределение данных**
    - Главный поток считывает параметры: максимальное количество потоков, исходный текст и образец для поиска
    - Вычисляется реальное количество потоков как минимум между максимальным количеством потоков и количеством возможных позиций для поиска
    - Текст разбивается на перекрывающиеся сегменты для каждого потока (перекрытие = длина образца - 1)
2. **Параллельный поиск**
    - Каждый поток выполняет наивный алгоритм поиска в своем сегменте текста
    - Наивный алгоритм последовательно проверяет каждую позицию в сегменте, сравнивая символы образца с соответствующими символами текста
    - Найденные позиции сохраняются в локальный массив каждого потока
3. **Синхронизация результатов**
    - После завершения поиска каждый поток добавляет свои результаты в общий массив

- Доступ к общему массиву защищается мьютексом для предотвращения гонки данных
- Главный поток ожидает завершения всех рабочих потоков

4. **Вывод результатов**
   - Программа выводит количество найденных вхождений, их позиции в тексте
   - Отображается время выполнения и фактическое количество использованных потоков
   - Предоставляется PID процесса для демонстрации потоков средствами операционной системы

Код программы

lab2_naive_search.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <string.h>
#include <time.h>
#include <unistd.h>

int MAX_THREADS = 4;

typedef struct {
    int id;
    char *text;
    char *pattern;
    int start;
    int end;
    int pattern_len;
    int *results;
    int *result_count;
    pthread_mutex_t *mutex;
} ThreadData;

pthread_t* threads;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void naive_search(const char *text, const char *pattern, int start, int end,
                  int pattern_len, int *positions, int *count) {
    for (int i = start; i <= end - pattern_len; i++) {
        int j;
        for (j = 0; j < pattern_len; j++) {
            if (text[i + j] != pattern[j]) break;
        }
        if (j == pattern_len) {
            positions[*count] = i;
            (*count)++;
        }
    }
```

```c
}

void* thread_function(void *arg) {
    ThreadData *data = (ThreadData *)arg;

    int local_count = 0;
    int *local_positions = malloc(sizeof(int) * (data->end - data->start));

    naive_search(data->text, data->pattern, data->start, data->end,
                 data->pattern_len, local_positions, &local_count);

    pthread_mutex_lock(data->mutex);
    for (int i = 0; i < local_count; i++) {
        data->results[*data->result_count] = local_positions[i];
        (*data->result_count)++;
    }
    pthread_mutex_unlock(data->mutex);

    free(local_positions);
    return NULL;
}

int main(int argc, char *argv[]) {
    if (argc != 4) {
        printf("Usage: %s <max_threads> <text> <pattern>\n", argv[0]);
        return 1;
    }

    MAX_THREADS = atoi(argv[1]);
    char *text = argv[2];
    char *pattern = argv[3];

    int text_len = strlen(text);
    int pattern_len = strlen(pattern);

    if (pattern_len > text_len) {
        printf("Pattern longer than text\n");
        return 1;
    }

    if (MAX_THREADS <= 0) {
        printf("Thread count must be positive\n");
        return 1;
    }

    int actual_threads = (text_len - pattern_len + 1 < MAX_THREADS) ?
                         text_len - pattern_len + 1 : MAX_THREADS;

    threads = malloc(sizeof(pthread_t) * actual_threads);
    ThreadData *thread_data = malloc(sizeof(ThreadData) * actual_threads);

    int *results = malloc(sizeof(int) * text_len);
    int result_count = 0;
```

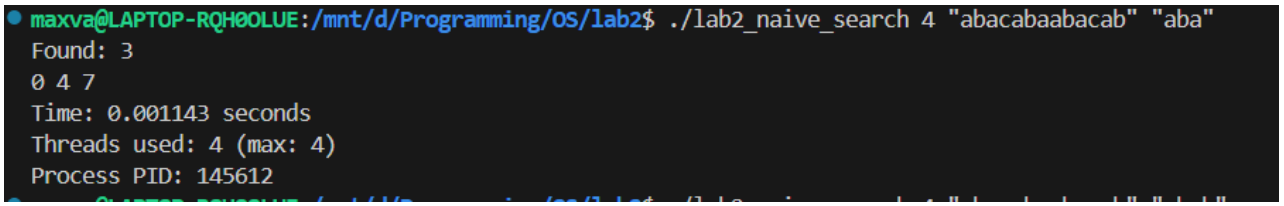
```c
    struct timespec start, end;
    clock_gettime(CLOCK_MONOTONIC, &start);

    int chunk_size = (text_len - pattern_len + 1) / actual_threads;
    int remainder = (text_len - pattern_len + 1) % actual_threads;

    int current_start = 0;
    for (int i = 0; i < actual_threads; i++) {
        thread_data[i].id = i;
        thread_data[i].text = text;
        thread_data[i].pattern = pattern;
        thread_data[i].start = current_start;

        int chunk = chunk_size + (i < remainder ? 1 : 0);
        thread_data[i].end = current_start + chunk + pattern_len - 1;
        if (thread_data[i].end > text_len) thread_data[i].end = text_len;

        thread_data[i].pattern_len = pattern_len;
        thread_data[i].results = results;
        thread_data[i].result_count = &result_count;
        thread_data[i].mutex = &mutex;

        current_start += chunk;

        pthread_create(&threads[i], NULL, thread_function, &thread_data[i]);
    }

    for (int i = 0; i < actual_threads; i++) {
        pthread_join(threads[i], NULL);
    }

    clock_gettime(CLOCK_MONOTONIC, &end);
    double time_sec = (end.tv_sec - start.tv_sec) + (end.tv_nsec - start.tv_nsec) / 1e9;

    printf("Found: %d\n", result_count);
    for (int i = 0; i < result_count; i++) {
        printf("%d ", results[i]);
    }
    printf("\n");

    printf("Time: %lf seconds\n", time_sec);
    printf("Threads used: %d (max: %d)\n", actual_threads, MAX_THREADS);
    printf("Process PID: %d\n", getpid());

    free(threads);
    free(thread_data);
    free(results);

    return 0;
}
```

benchmark.sh

```
#!/bin/bash

echo "Performance benchmark for multithreaded string search"
echo "====================================================="

gcc -o lab2_naive_search lab2_naive_search.c -lpthread

TEXT="abacabaabacababacabaabacababacabaabacababacabaabacaba"
PATTERN="aba"

echo "Text: $TEXT"
echo "Pattern: $PATTERN"
echo ""

for threads in 1 2 4 8
do
    echo "Threads: $threads"
    ./lab2_naive_search $threads "$TEXT" "$PATTERN" | grep -E "(Time:|Threads used:)"
    echo "---------------------------------------"
done
```

Протокол работы программы

Тестирование:



Strace:

```
$ strace -f ./lab2_naive_search 4 "abacabaabacab" "aba"

execve("./lab2_naive_search", ["./lab2_naive_search", "4", "abacabaabacab", "aba"],
0x7fff7aca41d0 /* 35 vars */) = 0

brk(NULL)                               = 0x5569a1daa000

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f1989aff000

access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

fstat(3, {st_mode=S_IFREG|0644, st_size=19935, ...}) = 0

mmap(NULL, 19935, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f1989afa000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832)
= 832
```

```
     pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784,
64) = 784

     fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0

     pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784,
64) = 784

     mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f19898e8000

     mmap(0x7f1989910000, 1605632, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7f1989910000

     mmap(0x7f1989a98000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x7f1989a98000

     mmap(0x7f1989ae7000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x7f1989ae7000

     mmap(0x7f1989aed000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f1989aed000

     close(3)                                = 0

     mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f19898e5000

     arch_prctl(ARCH_SET_FS, 0x7f19898e5740) = 0

     set_tid_address(0x7f19898e5a10)         = 210757

     set_robust_list(0x7f19898e5a20, 24)     = 0

     rseq(0x7f19898e6060, 0x20, 0, 0x53053053) = 0

     mprotect(0x7f1989ae7000, 16384, PROT_READ) = 0

     mprotect(0x55697c145000, 4096, PROT_READ) = 0

     mprotect(0x7f1989b37000, 8192, PROT_READ) = 0

     prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

     munmap(0x7f1989afa000, 19935)           = 0

     getrandom("\x51\xcd\x84\x02\x38\x64\x1b\x38", 8, GRND_NONBLOCK) = 8

     brk(NULL)                               = 0x5569a1daa000

     brk(0x5569a1dcb000)                     = 0x5569a1dcb000

     rt_sigaction(SIGRT_1, {sa_handler=0x7f1989981530, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f198992d330}, NULL, 8)
= 0

     rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

     mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x7f19890e4000

     mprotect(0x7f19890e5000, 8388608, PROT_READ|PROT_WRITE) = 0

     rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

     clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f19898e4990,
parent_tid=0x7f19898e4990, exit_signal=0, stack=0x7f19890e4000, stack_size=0x7fff80,
tls=0x7f19898e46c0}strace: Process 210758 attached
```

```
 => {parent_tid=[210758]}, 88) = 210758

[pid 210758] rseq(0x7f19898e4fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 210757] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 210758] <... rseq resumed>)        = 0

[pid 210757] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 210758] set_robust_list(0x7f19898e49a0, 24 <unfinished ...>

[pid 210757] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

[pid 210758] <... set_robust_list resumed>) = 0

[pid 210757] <... mmap resumed>)        = 0x7f19888e3000

[pid 210758] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 210757] mprotect(0x7f19888e4000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

[pid 210758] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 210758] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0
<unfinished ...>

[pid 210757] <... mprotect resumed>)    = 0

[pid 210758] <... mmap resumed>)        = 0x7f19808e3000

[pid 210757] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>

[pid 210758] munmap(0x7f19808e3000, 57790464 <unfinished ...>

[pid 210757] <... rt_sigprocmask resumed>[], 8) = 0

[pid 210757]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_S
ETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f19890e3990,
parent_tid=0x7f19890e3990, exit_signal=0, stack=0x7f19888e3000, stack_size=0x7fff80,
tls=0x7f19890e36c0} <unfinished ...>

[pid 210758] <... munmap resumed>)      = 0

[pid 210758] munmap(0x7f1988000000, 9318400strace: Process 210759 attached

 <unfinished ...>

[pid 210757] <... clone3 resumed> => {parent_tid=[210759]}, 88) = 210759

[pid 210758] <... munmap resumed>)      = 0

[pid 210757] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 210759] rseq(0x7f19890e3fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 210757] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 210758] mprotect(0x7f1984000000, 135168, PROT_READ|PROT_WRITE <unfinished ...>

[pid 210757] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

[pid 210759] <... rseq resumed>)        = 0

[pid 210757] <... mmap resumed>)        = 0x7f19880e2000

[pid 210758] <... mprotect resumed>)    = 0
```

```
[pid 210757] mprotect(0x7f19880e3000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid 210759] set_robust_list(0x7f19890e39a0, 24 <unfinished ...>
[pid 210757] <... mprotect resumed>)    = 0
[pid 210759] <... set_robust_list resumed>) = 0
[pid 210758] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid 210757] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid 210759] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 210757] <... rt_sigprocmask resumed>[], 8) = 0
[pid 210758] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 210757] clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f19888e2990, parent_tid=0x7f19888e2990, exit_signal=0, stack=0x7f19880e2000, stack_size=0x7fff80, tls=0x7f19888e26c0} <unfinished ...>
[pid 210759] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 210758] madvise(0x7f19890e4000, 8368128, MADV_DONTNEED <unfinished ...>
[pid 210759] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid 210758] <... madvise resumed>)     = 0
strace: Process 210760 attached
[pid 210757] <... clone3 resumed> => {parent_tid=[210760]}, 88) = 210760
[pid 210759] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 210758] exit(0 <unfinished ...>
[pid 210757] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 210760] rseq(0x7f19888e2fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 210759] madvise(0x7f19888e3000, 8368128, MADV_DONTNEED <unfinished ...>
[pid 210757] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 210760] <... rseq resumed>)        = 0
[pid 210758] <... exit resumed>)        = ?
[pid 210757] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
[pid 210760] set_robust_list(0x7f19888e29a0, 24 <unfinished ...>
[pid 210759] <... madvise resumed>)     = 0
[pid 210760] <... set_robust_list resumed>) = 0
[pid 210758] +++ exited with 0 +++
[pid 210759] exit(0 <unfinished ...>
[pid 210757] <... mmap resumed>)        = 0x7f19837ff000
[pid 210760] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 210757] mprotect(0x7f1983800000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
```

```
[pid 210759] <... exit resumed>)        = ?

[pid 210757] <... mprotect resumed>)    = 0

[pid 210760] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 210759] +++ exited with 0 +++

[pid 210760] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

[pid 210760] madvise(0x7f19880e2000, 8368128, MADV_DONTNEED) = 0

[pid 210760] exit(0)                    = ?

[pid 210760] +++ exited with 0 +++

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|C
LONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f1983fff990,
parent_tid=0x7f1983fff990, exit_signal=0, stack=0x7f19837ff000, stack_size=0x7fff80,
tls=0x7f1983fff6c0}strace: Process 210761 attached

 => {parent_tid=[210761]}, 88) = 210761

[pid 210761] rseq(0x7f1983ffffe0, 0x20, 0, 0x53053053) = 0

[pid 210757] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 210761] set_robust_list(0x7f1983fff9a0, 24 <unfinished ...>

[pid 210757] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 210761] <... set_robust_list resumed>) = 0

[pid 210757] futex(0x7f1983fff990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 210761,
NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 210761] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

[pid 210761] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

[pid 210761] madvise(0x7f19837ff000, 8368128, MADV_DONTNEED) = 0

[pid 210761] exit(0)                    = ?

[pid 210757] <... futex resumed>)       = 0

[pid 210761] +++ exited with 0 +++

fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x7), ...}) = 0

write(1, "Found: 3\n", 9Found: 3

)               = 9

write(1, "0 4 7 \n", 70 4 7

)                 = 7

write(1, "Time: 0.003325 seconds\n", 23Time: 0.003325 seconds

) = 23

write(1, "Threads used: 4 (max: 4)\n", 25Threads used: 4 (max: 4)

) = 25

getpid()                                = 210757
```

```
write(1, "Process PID: 210757\n", 20Process PID: 210757
)   = 20
exit_group(0)                           = ?
+++ exited with 0 +++
```

Потоков	Время	Ускорение	Эффективность
1	0.000110	1	1
2	0.000097	1,13	0,57
4	0.000462	0,24	0,06
8	0.000746	0,15	0,02

Эксперимент показал, что при малом объеме входных данных использование многопоточности неэффективно. Наилучшее время достигнуто на 2 потоках (ускорение 1.13), однако дальнейшее увеличение числа потоков приводит к замедлению из-за преобладания накладных расходов над полезной работой. Для демонстрации преимуществ параллелизма необходим больший объем вычислений.

Вывод

В ходе работы освоены основы многопоточного программирования: создание потоков через pthread, синхронизация мьютексами и распределение работы. На практике подтверждено, что эффективность многопоточности зависит от объема данных - при малых задачах накладные расходы превышают выгоду от параллелизма. Получен важный опыт оптимизации многопоточных алгоритмов.